pomponio_alessandro_2020-02

February 5, 2021

1 Alessandro Pomponio - 0000920265

[42]:
```
# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

# Variables
file_name = 'online_retail_red.txt'
separator = ';'
random_state = 42

# Directives
%matplotlib inline
np.random.seed(random_state)
```

1.1 1. Read a transactions database in sparse format from a txt file into a list of lists. The file contains one transaction per line, each line has a variable number of items, separated by a separator. The external list will have a list for each transaction, each transaction is a list of items. Show the first two transactions (3 points)

We start by reading the database from the text file into a list of lists

[2]:
```
# Open the file
file = open(file_name, mode = 'r')

# The transactions are separated by a ';'
# We must also strip the '\n' at the end of the transaction
transactions = [line.strip('\n').split(separator) for line in file.readlines()]
```

Showing the first two transactions:

[3]:
```
print(f"First transaction:\t{transactions[0]}\n\nSecond transaction:
↪\t{transactions[1]}")
```

```
First transaction:      ['CREAM CUPID HEARTS COAT HANGER', 'GLASS STAR FROSTED
T-LIGHT HOLDER', 'KNITTED UNION FLAG HOT WATER BOTTLE', 'RED WOOLLY HOTTIE WHITE
HEART.', 'SET 7 BABUSHKA NESTING BOXES', 'WHITE HANGING HEART T-LIGHT HOLDER',
'WHITE METAL LANTERN']

Second transaction:     ['HAND WARMER RED POLKA DOT', 'HAND WARMER UNION JACK']
```

1.2 2. Encode the list of lists into a binary representation and transform it into a dataframe whose columns are the items show the head of that dataframe

To perform this task, we will use a transaction encoder to transform the data in a binary representation

[4]:
```python
# Encode the transactions
encoder = TransactionEncoder()
encoded_transactions = encoder.fit_transform(transactions)

# Put the data in a dataframe
df = pd.DataFrame(encoded_transactions.astype(int), columns = encoder.columns_)
df.head()
```

[4]:
```
   10 COLOUR SPACEBOY PEN  12 COLOURED PARTY BALLOONS  \
0                       0                           0
1                       0                           0
2                       0                           0
3                       0                           0
4                       0                           0

   12 DAISY PEGS IN WOOD BOX  12 EGG HOUSE PAINTED WOOD  \
0                          0                          0
1                          0                          0
2                          0                          0
3                          0                          0
4                          0                          0

   12 IVORY ROSE PEG PLACE SETTINGS  12 MESSAGE CARDS WITH ENVELOPES  \
0                                 0                                0
1                                 0                                0
2                                 0                                0
3                                 0                                0
4                                 0                                0

   12 PENCIL SMALL TUBE WOODLAND  12 PENCILS SMALL TUBE RED RETROSPOT  \
0                              0                                    0
1                              0                                    0
2                              0                                    0
3                              0                                    0
4                              0                                    0
```

```
   12 PENCILS SMALL TUBE SKULL  12 PENCILS TALL TUBE POSY  …  \
0                           0                          0  …
1                           0                          0  …
2                           0                          0  …
3                           0                          0  …
4                           0                          0  …

   ZINC HEART LATTICE T-LIGHT HOLDER  ZINC HEARTS PLANT POT HOLDER  \
0                                  0                             0
1                                  0                             0
2                                  0                             0
3                                  0                             0
4                                  0                             0

   ZINC HERB GARDEN CONTAINER  ZINC METAL HEART DECORATION  \
0                           0                            0
1                           0                            0
2                           0                            0
3                           0                            0
4                           0                            0

   ZINC SWEETHEART SOAP DISH  ZINC SWEETHEART WIRE LETTER RACK  \
0                          0                                 0
1                          0                                 0
2                          0                                 0
3                          0                                 0
4                          0                                 0

   ZINC T-LIGHT HOLDER STAR LARGE  ZINC T-LIGHT HOLDER STARS SMALL  \
0                               0                                0
1                               0                                0
2                               0                                0
3                               0                                0
4                               0                                0

   ZINC WILLIE WINKIE  CANDLE STICK  ZINC WIRE SWEETHEART LETTER TRAY
0                                 0                                 0
1                                 0                                 0
2                                 0                                 0
3                                 0                                 0
4                                 0                                 0

[5 rows x 2968 columns]
```

1.3 3. Find a value of min_support such that the apriori algorithm generates at least 8 frequent itemsets with at least 2 items. Output the result with the message below (5 points)

min_support: 0.xxxx - number of itemsets with at least 2 items: nn

In order to perform the task, we will specify a range of "reasonable" `min_support` values and iterate until we meet our criteria

```
[21]: # Requirements
min_itemsets = 8
min_item_in_itemset = 2

# "Reasonable" range
support_range = np.arange(0.1, 0.01, -0.01)
```

```
[24]: min_support = 0

for s_value in support_range:

    print(f"Trying support value {s_value:.2f}")

    frequent_itemsets = apriori(df, min_support = s_value, use_colnames = True)

    # Calculate the number of itemsets that contain at least␣
↪`min_item_in_itemset` items
    itemsets_above_threshold = sum([len(itemset) >= min_item_in_itemset for␣
↪itemset in frequent_itemsets.itemsets])

    if itemsets_above_threshold >= min_itemsets:
        min_support = s_value
        break

if min_support == 0:
    print("No itemset found! Try again with a bigger range!")
else:
    print(f"I've selected min_support = {min_support:.2f}, which produced␣
↪{len(frequent_itemsets)} itemsets, {itemsets_above_threshold} of which had␣
↪more than {min_item_in_itemset} items")
```

```
Trying support value 0.10
Trying support value 0.09
Trying support value 0.08
Trying support value 0.07
Trying support value 0.06
Trying support value 0.05
Trying support value 0.04
Trying support value 0.03
I've selected min_support = 0.03, which produced 148 itemsets, 12 of which had
```

```
more than 2 items
```

Output as requested

```
[26]: print(f"min_support: {min_support:.4f} - number of itemsets with at least 2␣
      ↪items: {itemsets_above_threshold}")
```

```
min_support: 0.0300 - number of itemsets with at least 2 items: 12
```

1.4 4. Find the minimum metric threshold such that at least 10 association rules are extracted from the frequent itemsets found (5 points)

use "confidence" as metric and output the line below:

Metric: "confidence" - min_metric: 0.xxxx - Number of rules: n

Just like we did before, let us create a "reasonable" range to iterate over

```
[28]: # Threshold
      min_rules = 10

      # "Reasonable" range
      confidence_range = np.arange(1, 0.01, -0.01)
```

```
[29]: min_confidence = 0

      for c_value in confidence_range:

          print(f"Trying confidence value {c_value:.2f}")
          rules = association_rules(frequent_itemsets, metric = "confidence",␣
      ↪min_threshold = c_value)

          if len(rules) >= min_rules:
              min_confidence = c_value
              break

      if min_confidence == 0:
          print("No rule set met the requirements! Try again with a bigger range!")
      else:
          print(f"I've selected min_confidence = {min_confidence:.2f}, which produced␣
      ↪{len(rules)} rules")
```

```
Trying confidence value 1.00
Trying confidence value 0.99
Trying confidence value 0.98
Trying confidence value 0.97
Trying confidence value 0.96
Trying confidence value 0.95
Trying confidence value 0.94
Trying confidence value 0.93
Trying confidence value 0.92
Trying confidence value 0.91
Trying confidence value 0.90
Trying confidence value 0.89
Trying confidence value 0.88
Trying confidence value 0.87
Trying confidence value 0.86
Trying confidence value 0.85
Trying confidence value 0.84
Trying confidence value 0.83
Trying confidence value 0.82
Trying confidence value 0.81
Trying confidence value 0.80
Trying confidence value 0.79
Trying confidence value 0.78
Trying confidence value 0.77
Trying confidence value 0.76
Trying confidence value 0.75
Trying confidence value 0.74
Trying confidence value 0.73
Trying confidence value 0.72
Trying confidence value 0.71
Trying confidence value 0.70
Trying confidence value 0.69
Trying confidence value 0.68
Trying confidence value 0.67
Trying confidence value 0.66
Trying confidence value 0.65
Trying confidence value 0.64
Trying confidence value 0.63
Trying confidence value 0.62
Trying confidence value 0.61
I've selected min_confidence = 0.61, which produced 11 rules
```

Print out as required:

[31]:
```
print(f'Metric: "confidence" - min_metric: {min_confidence:.4f} - Number of␣
↪rules: {len(rules)}')
```

```
Metric: "confidence" - min_metric: 0.6100 - Number of rules: 11
```

1.5 5. Print the first 10 rules found, sorted by descending confidence and support (3 points)

We now know that there are 11 rules, so we can just limit the output.

It is important to drop the index when sorting the rules so that a new one can be built

```
[37]: sorted_rules = rules.sort_values(by = ['confidence', 'support'], ascending =␣
       ↪False).reset_index(drop=True)
      sorted_rules[:-1]
```

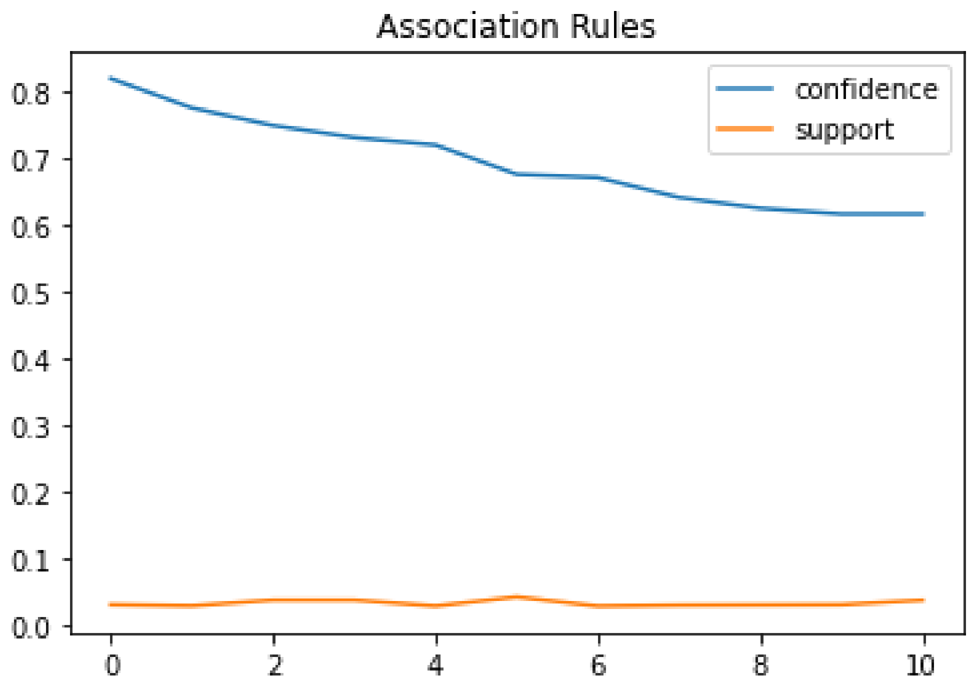
```
[37]:                            antecedents                             consequents  \
      0      (PINK REGENCY TEACUP AND SAUCER)   (GREEN REGENCY TEACUP AND SAUCER)
      1      (PINK REGENCY TEACUP AND SAUCER)   (ROSES REGENCY TEACUP AND SAUCER)
      2     (GREEN REGENCY TEACUP AND SAUCER)   (ROSES REGENCY TEACUP AND SAUCER)
      3     (ROSES REGENCY TEACUP AND SAUCER)   (GREEN REGENCY TEACUP AND SAUCER)
      4   (GARDENERS KNEELING PAD CUP OF TEA)  (GARDENERS KNEELING PAD KEEP CALM)
      5             (JUMBO BAG PINK POLKADOT)             (JUMBO BAG RED RETROSPOT)
      6   (PAPER CHAIN KIT VINTAGE CHRISTMAS)     (PAPER CHAIN KIT 50'S CHRISTMAS)
      7          (ALARM CLOCK BAKELIKE GREEN)            (ALARM CLOCK BAKELIKE RED)
      8      (JUMBO  BAG BAROQUE BLACK WHITE)             (JUMBO BAG RED RETROSPOT)
      9     (GREEN REGENCY TEACUP AND SAUCER)    (PINK REGENCY TEACUP AND SAUCER)

         antecedent support  consequent support   support  confidence       lift  \
      0            0.039331            0.052255  0.032282    0.820768  15.707077
      1            0.039331            0.053541  0.030547    0.776671  14.505982
      2            0.052255            0.053541  0.039219    0.750535  14.017835
      3            0.053541            0.052255  0.039219    0.732497  14.017835
      4            0.041960            0.050241  0.030267    0.721333  14.357586
      5            0.064843            0.108426  0.043919    0.677308   6.246751
      6            0.044758            0.062941  0.030100    0.672500  10.684680
      7            0.049010            0.052031  0.031498    0.642694  12.352165
      8            0.050912            0.108426  0.031890    0.626374   5.776988
      9            0.052255            0.039331  0.032282    0.617773  15.707077

         leverage  conviction
      0  0.030226    5.287817
      1  0.028441    4.237964
      2  0.036421    3.793958
      3  0.036421    3.542939
      4  0.028159    3.408228
      5  0.036888    2.762927
      6  0.027283    2.861250
      7  0.028948    2.653102
      8  0.026370    2.386273
      9  0.030226    2.513347
```

1.6 6. Plot confidence and support for all the sorted rules found (3 points)

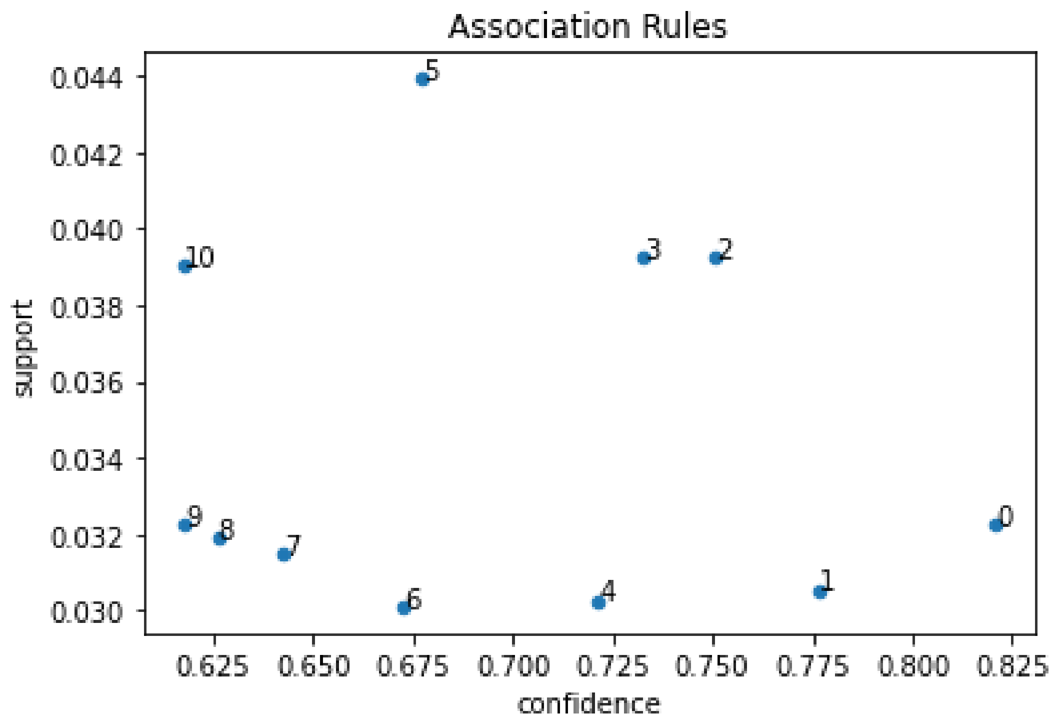
```
[38]: sorted_rules[['confidence','support']].plot(title='Association Rules');
```



1.7 7. Scatter plot the rules by confidence and support, labelling the points with the index value of the corresponding rule (3 points)

[57]:
```
# Create a scatter plot of the sorted rules
fig = sorted_rules.plot.scatter(x = 'confidence', y = 'support', title =␣
↪'Association Rules');

# Iterate over all the rules and annotate them with their index
for i in range(len(sorted_rules)):
    fig.annotate(text = i, xy = (sorted_rules['confidence'][i],␣
↪sorted_rules['support'][i]))
```


Association Rules
support
confidence
0.044
0.042
0.040
0.038
0.036
0.034
0.032
0.030
0.625
0.650
0.675
0.700
0.725
0.750
0.775
0.800
0.825
0
1
2
3
4
5
6
7
8
9
10